Prof. Dr. A. Podelski, Dr. B. Westphal
S. Feo Arenis

Sommersemester 2015

Softwaretechnik/Software Engineering

http://swt.informatik.uni-freiburg.de/teaching/SS2015/swtvl

Exercise Sheet 3

Early submission: Friday, 2015-05-29, 12:00 Regular submission: Monday, 2015-06-01, 14:00

Exercise 1 – (Requirements Elicitation) (10/20 Points)

At the *Softwarepraktikum*, one requirement from the customer is the following:

> "Das Spiel muss mindestens 1000 gleichzeitig aktive Spielobjekte handhaben können. Es müssen sich jedoch nicht so viele Spielobjekte im fertigen Spiel befinden, allerdings muss mindestens eine Tech-Demo vorhanden sein, die bei Bedarf die gewünschte Funktionalität zeigt."
>
> "The game must be able to handle at least 1000 simultaneous active game objects. There need not be that many objects in the completed game, however, at least a tech-demo must be provided that demonstrates this functionality whenever required."

In this exercise, you are asked to clarify and specify that requirement as precise as possible, i.e. to assume the role of an analyst or requirements engineer:

(i) State what terms or conditions need to be clarified. Justify each statement by giving at least two (at best: plausible) **interpretations** of the requirement that highlight the ambiguity of that term or condition. (2)

Hint: in our opinion, there are at least 5 terms which need further investigation. Do you find more?

Bonus: For the 6th to 10th term or condition you justify that clarification is necessary, you get one bonus point. (5 Bonus)

(ii) Give a list of **questions** you would ask the customer in order to clarify the terms and conditions you found in (i). For each question, explain what information you expect to receive, or which interpretation are you trying to refute or confirm.

Ask the questions to the customer* and document the answers received. If you are not satisfied with the answer, reformulate your question and ask again; elicit the exact conditions. (2)

∗*: For the course of this exercise, your tutor has been hired by the* Softwarepraktikum *team to represent the customer in the requirements analysis. So ask your questions to your software engineering tutor by mail, your tutor's answer is authoritative; if necessary, your tutor will clarify with the "big bosses" of* Softwarepraktikum *if needed. Use the information obtained to complete the rest of this exercise tasks.*

(iii) Create a **dictionary** that defines the *terms* that you consider necessary to completely specify the requirement. For each new term not already discussed in task (i), briefly discuss why is it necessary to include it in the dictionary, i.e., which interpretations of the term that are not desirable are avoided. For the definitions, use the information obtained from the previous task. (2)

(iv) Based on the information collected, give a **specification** for the requirement. The specification should be as precise as possible. (2)

Hint: Decision tables seem not to be a very good choice to represent your specification. You may use as much mathematical notation as you deem necessary.

(v) Justify why your specification is relevant, complete, consistent, neutral, comprehensible and testable. (2)

Exercise 2 – (Analysis of Decision Tables) (5/20 Points)

Consider the following decision table for the controller of the power windows of a car:

DT: Power window		**R1**	**R2**	**R3**	**R4**	**R5**
C1	Button UP	×	*	*	*	-
C2	Button DOWN	*	×	*	*	-
C3	Top reached	-	*	×	*	*
C4	Bottom reached	*	-	*	×	*
A1	Move window up	×	-	-	-	-
A2	Move window down	-	×	-	-	-
A3	Stop window	-	-	-	×	×
A4	Auto full close	-	-	×	-	-

Conflict axiom: (Top reached $\wedge$ Bottom reached)

(Models that in the real-world, top and bottom are never reached at the same time.)

(i) Is the decision table **complete**? Justify your answer: If it is complete, state why; if it is not, give examples of missing conditions. (1)

(ii) Is the decision table **deterministic**? Justify your answer: If it is deterministic, state why; if it is not, state which rules are non-deterministic. (1)

(iii) Is the decision table **consistent** in the collecting semantics? Justify your answer: If it is consistent, state why; if it is not, state which rules are conflicting and propose a fix (possibly involving the conflict axiom). (2)

(iv) Which rules would be inconsistent if the conflict clause for "Top reached" and "Bottom reached" were removed? What undesirable situation would be caused by the inconsistency? Is the inconsistency problematic in the *collecting semantics*, in the *interleaving semantics*, or both? (1)

Exercise 3 – (Creation of Decision Tables) (5/20 Points)

In this exercise, requirements for the software of an electric car with range extender need to be specified. The car drives using only electric power but carries a generator on board that burns regular fuel.
The customer states the following:

- "When the battery level drops below 15%, the generator is started automatically"
- "The generator cannot be started if the fuel tank is empty"

- "There is a switch that overrides automatic charging: It can be set to `OFF`, to prevent the generator from starting; to `ON`, to start the generator at any time, and to `AUTO`, to allow the system to start and stop the generator according to the state of the battery"
- "If the battery charge reaches 100%, the generator must be turned off"

Considering those informal requirements:

(i) Specify the requirements as **rules** by creating a complete, consistent **decision table** in terms of conditions, actions and conflict axioms you identified.

Give a computation path induced by your table that would be useful to convince the customer that your specification is useful. The computation path should depict how system behaviors that are considered desirable by the customer are induced by your table. (3)

(ii) The informal requirements are ambiguous: The switch can be turned to `ON` even when the battery is full. Does this ambiguity lead to conflicting actions in your specification if interpreted with *collecting semantics*? Does the conflict resolve by switching to *interleaving semantics*, and if yes, do you think the resulting computation paths are expected by the customer? (2)